The Report committee for Bruce A. Zabava

Certifies that this is the approved version of the following report:

Removable Media Software Engineering

Approved by

Supervising Committee:

_____

Sarfraz Khurshid, Supervisor

_____

K. Suzanne Barber

Removable Media Software Engineering

by

Bruce A. Zabava, B.S.

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

The University of Texas at Austin

December 2010

Abstract

Removable Media Software Engineering

by

Bruce A. Zabava, M.S.E.

The University of Texas at Austin, 2010

Supervisor: Sarfraz Khurshid

This report focuses on software solutions that are fully contained on CD, DVD, or USB storage devices. When the removable media is selected as the boot device on a computer system, it will launch its own operating system and then execute the actions it was written to perform. Organizations often have many types of removable media solutions that are engineered to perform various tasks; however, this report will focus on only two types of removable media projects that are often needed by original equipment manufacturers (OEMs): the *end user reinstallation media* (EURM) and the *field service reinstallation media* (FSRM).

In this document we explain the purpose of EURM and FSRM solutions, the engineering methods used to create them, what engineering measurements can be used to achieve project goals, what to consider when implementing process improvements, and finally how to analyze the results of said process improvements.

Table of Contents

List of Tables

List of Illustrations

Chapter 1: Introduction

Software solutions are often designed to run from removable media that can be used on multiple computer systems to perform certain tasks or actions. The removable media solutions described in this report are all self-sufficient in scope; they do not require another operating system to be up and running in order to work. When the removable media is selected as the boot device on a computer system, it will launch its own operating system and then execute the actions it was written to perform. These software solutions are often used for installing a new operating system onto a target system or for modifying target system software without having to actually boot into the currently existing operating system. It is the intent of this paper to describe the engineering considerations used when developing removable media software solutions.

Chapter 2: Removable Media Survey

The removable media software engineering solutions covered by this report include the *End User Reinstallation Media* (EURM) and the *Field Service Reinstallation Media* (FSRM) [1]. The following sections in this chapter will describe each of these solutions in more detail.

End User Reinstallation Media (EURM)

The *end user reinstallation media* (EURM) is typically used by an individual to re-install a computer system's operating system. This media normally comes in the form of a compact disc (CD) for Microsoft Windows XP operating systems and in the form of a digital versatile disc (DVD) for Microsoft Windows Vista or Microsoft Windows 7 operating systems. Due to fewer computers including DVD drive readers as their form factors shrink in size, the industry has recently started to create and release EURM on universal serial bus (USB) flash drives.

This EURM is triggered to run by inserting the solution's media into an appropriate disc drive reader or USB port and then booting the computer system to the target media. This step often requires the end user to enter into their system's boot order menu or their system's basic input/output system (BIOS) in order to trigger the EURM to boot instead of the computer system's current operating system.

An end user will often resort to using this media when their current operating system is no longer functioning to their expectations or when they purchase a new hard disk drive (HDD) to replace the drive containing their current operating system software.

The primary goals the engineers consider when creating these types of solutions are: (1) to minimize service calls and (2) to reduce procurement costs. In order to minimize service calls, the solution is designed to flow in a standardized manner with no

complex menus or choices. The engineer will often try to adopt a typical Microsoft operating system installation event flow and will remove any advanced user options that might generate too much confusion.

In order to reduce procurement costs, the engineer will attempt to build their solutions using the media format that costs the least to mass produce. Currently, USB flash drives are the most expensive format to mass produce and are avoided in EURM solution scenarios unless absolutely necessary. A USB flash drive will only be used in one of two scenarios: (1) the system supported does not have a CD/DVD drive reader or (2) the solution being produced will not fit onto a 8.5 GB DVD.

The next most expensive format currently used in this industry is the single-sided double-layered DVD format (DVD-9), followed by the standard, single-sided single-layer, DVD format (DVD-5). The DVD-9 can hold solutions of up to 8.5 GB in total size and the DVD-5 can hold solutions of up to 4.7 GB. Both DVD formats require the user have a DVD drive reader in order to use them.

The least expensive format to mass produce is the CD. CDs are often restricted to 650 MB because some of the vendors that mass produce these CDs will not guarantee data integrity for anything larger.

COST	MEDIA TYPE	CAPACITY
Most Expensive	USB	8 GB +
More Expensive	DVD-9	8.5 GB
Average	DVD-5	4.7 GB
Least Expensive	CD	650 MB

Table 1: Media Costs and Sizes

EURM projects are normally initiated by platform development teams when they begin to review service and support options for new computer system platforms. If no reinstallation media solution exists for the system being developed, then the

development manager will make a formal request to the software engineering team to design a solution.

The software engineers currently build EURM as generic, cross platform capable media which helps to reduce part complexity, which in turn, reduces overall business costs. Reducing the number of media parts to maintain not only helps in the procurement and image management chains, but also helps the actual factories manage their production lines in terms of where to store and distribute the media. As an example, if a factory builds 10 different platforms, and had 10 different articles of EURM to support the different platforms, they would need 10 different bins to store the articles of media instead of 1 bin. If the company offers different operating system SKUs to its customers, like Windows 7 Ultimate, Windows 7 Home Premium, and Windows 7 Starter, the number of bins needing to be maintained along the factory build line (which is often already short on space) would be 30. So, although it might make sense from a customer point of view to have a system image contained on the EURM instead of a generic operating system install, the cost to have system specific image installation media is very high.

The software engineers will often follow two different paths when creating EURM, depending on the operating system and platform(s) the solution is being designed for. The first path (using a Microsoft standard installation flow) is to take a Microsoft's original operating system installation media image and modify it so that it contains some original equipment manufacturer (OEM) customizations. These customizations are typically minor in nature and often only include adding OEM logos and graphics, OEM technical support contact information, OS activation logic, and the integration of Microsoft updates that a customer would have had to install from Windows Update immediately after installing their operating system. With Windows XP media, the software engineers will also include mass storage device drivers in order for

the media to get through Microsoft Windows XP text-mode setup successfully without the need for a floppy disk drive. As an additional step with Microsoft Windows Vista and Microsoft Windows 7 media, the software engineer will often modify the media to include as many language options as legally (and physically) possible. Illustration 2 and Illustration 3 are high level process flow diagrams for the creation of EURM that will use a standard Microsoft installation flow.

The second path the engineers might follow, when Microsoft does not have a user friendly operating system installation path, is to capture and deploy a generic operating system image using internally created image extraction tools. In this process, an engineer will generate a generic image for use on the reinstallation media and the media will then be designed to remove as much confusion as possible for the end user who might need to use the solution. The end user steps are often reduced to (1) insert and boot to the EURM, (2) have the end user acknowledge that they have the legal right to install the operating system onto the target computer system, (3) warn the user that all data on the hard drive will be lost, (4) deploy the image to the hard drive with some sort of background graphic that indicates to the user that the extraction process is making progress (see Illustration 1).

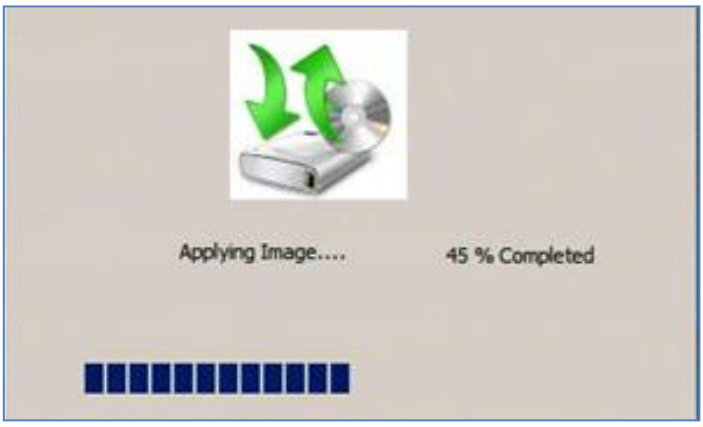

Illustration 1: Applying Image

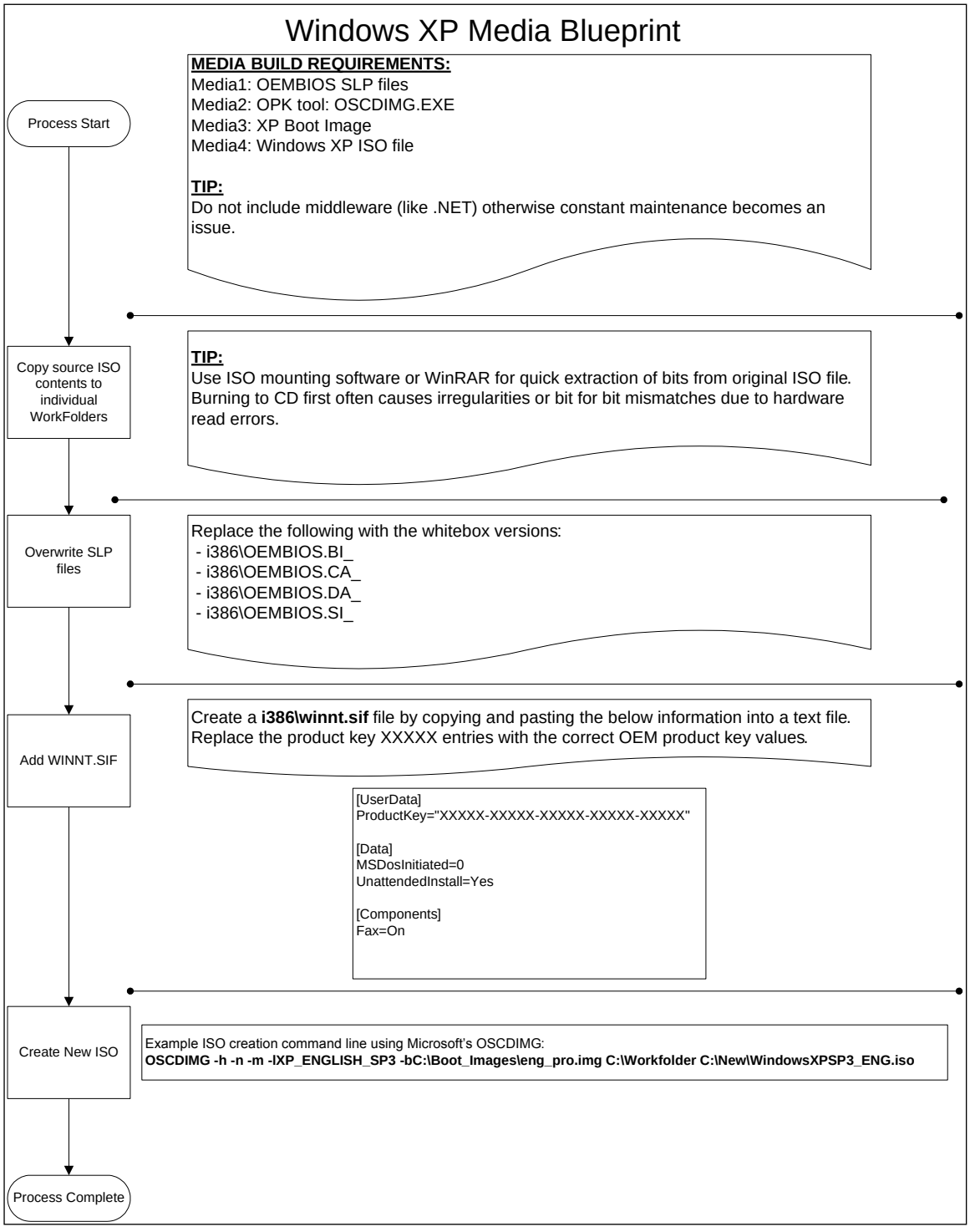

Illustration 2: Windows XP Media creation process

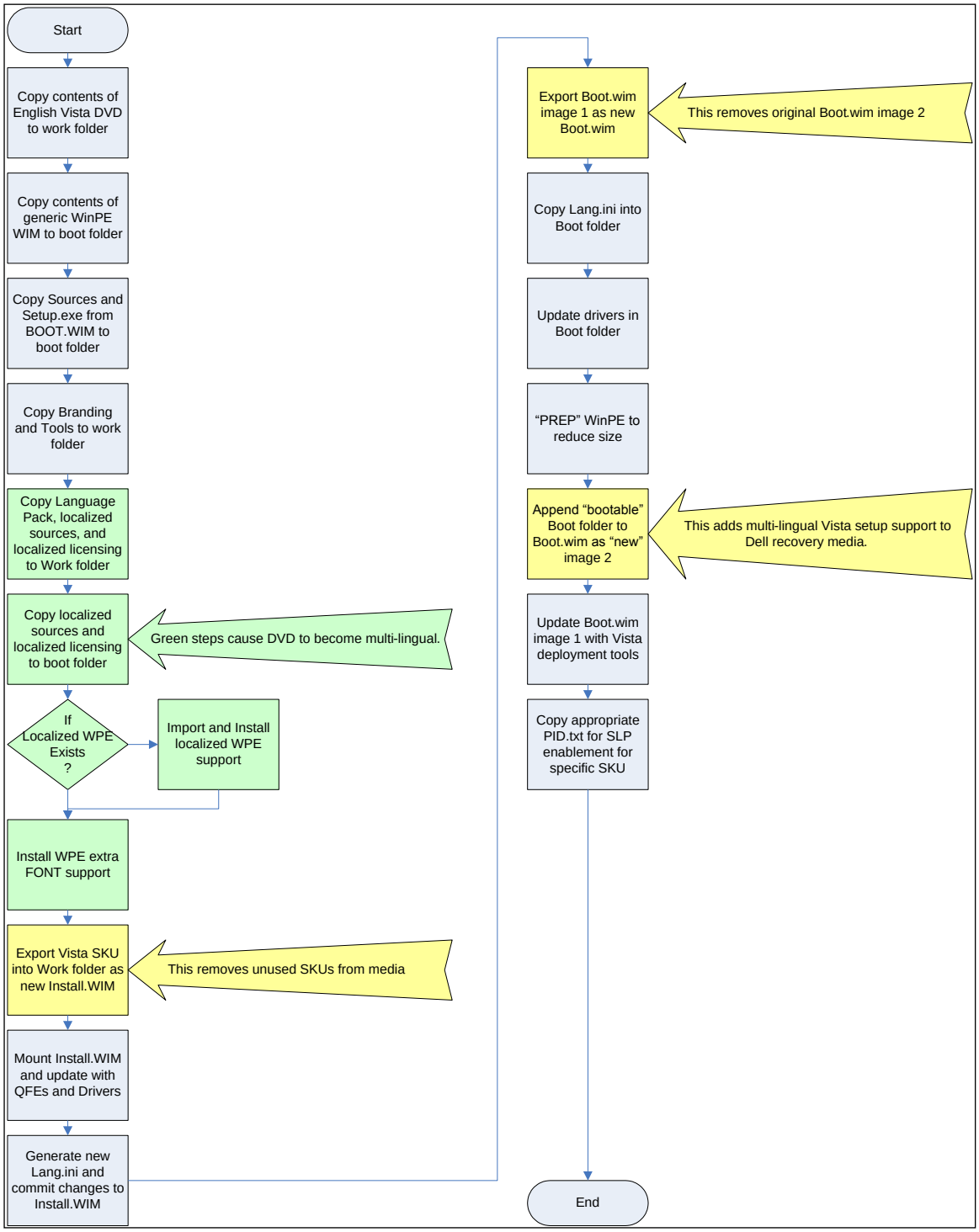

Illustration 3: Windows Vista Media creation process

Field Service Reinstallation Media (FSRM)

The *field service reinstallation media* (FSRM) is built to give field technicians the ability to reinstall a customer's operating system after a new hard disk drive is installed into a computer system. This media normally comes in the form of a universal serial bus (USB) hard disk drive (HDD), or USB HDD. Since USB HDDs can come in very large sizes, costs will be high, but the field technician can maintain a large library of operating system versions and language options on a single USB device.

The primary purpose of the FSRM is to give field techs the ability to quickly install the appropriate operating system onto the customer's computer system. The engineering goal is to reduce customer visit cycle times… the faster the field technician can get a customer up and running, the more customer visits the field technician can perform each day.

FSRM requests are normally initiated by internal Service and Support teams. When new operating systems come out, and their install times are taking too long using conventional installation methods, the services team will often ask the engineering team to help come up with solutions to reduce install times, which will in turn help reduce customer site visit cycle times.

The FSRM we will work with is currently designed to boot into a Windows Preinstallation Environment (WPE) [2] that automatically launches a HTML Application (HTA) front end. This graphic user interface (GUI) allows the field technician to quickly choose which operating system, version, and language they want to install. The tool then prepares the customer's hard disk drive and extracts the selected operating system image onto the drive.

The images placed on the FSRM are currently gathered directly from the original equipment manufacturer (OEM) factory servers. The OEM images often contain many

customizations, including Windows updates and middleware, like the .Net framework that would not typically be included with the default Microsoft image. Using these images greatly reduces customer site visit cycle times and, since the FSRM uses USB technology, allows the field technician to install the operating system even if the customer's computer does not have a working CD or DVD drive.

Chapter 3: Removable Media Measurements

Gathering appropriate measurements should be considered a fundamental step when engineering removable media solutions. With accurate measurement gathering techniques in place, the engineering team can better predict durations and results. Implementing measurements on the various build process tasks, will also allow the engineers to implement improvements in the way they create and manage their removable media solutions.

End User Reinstallation Media (EURM) Measurements

Since the primary goal when engineering EURM solutions is to minimize service calls by reducing complexity and to reduce business costs wherever possible [1]; the questions that need to be answered are typically, "*how do we measure usability?*", and "*how do we measure business costs?*".

Usability can be measured by counting the number of added deviations from standard Microsoft operating system installation flow that the customer might have to traverse to successfully install their operating system. This can be done by outlining the normal Microsoft operating system installation flow and then comparing the engineering solution installation flow to this standard. We hypothesize that the higher the added deviation count, the more complex the solution is to use and the greater the chance the customer might pick up the phone to call tech support with questions. (The hope here is that Microsoft has already taken the time to perform a usability study with end users of various skill levels to come up with the best installation experience possible for the operating system being installed.)

Screen	Microsoft solution	OEM solution	Usability
1	*-bypassed-*	Pick Setup Language	1
2	Pick OS Language	Pick OS language	1
3	Begin installation	Begin installation	1
4	EULA	EULA	1
5	Installation Type	Installation Type	1
6	Hard Drive selection	Hard Drive selection	1
7	Installing Windows Progress	Installing Windows Progress	1
8	User Name	User Name	1
9	Password	Password	1
10	Product Key	*-bypassed-*	0
11	Protect Your Computer	Protect Your Computer	0
12	Date & Time	Date & Time	0

Table 2: Media Installation Usability Count

From Table 2, it can be seen that that OEM solution is slightly more complex than Microsoft's default installation media when the process is first initialized. In this example, the OEM media includes many languages while the Microsoft solution only includes one language; screen 1 only applies to the OEM media. However, the OEM removes the product key window (screen 10) which reduces the usability factor score back to 0 over standard Microsoft installation process flow.

As a side note, internal call log audits showed that the number one reason why customers might call tech support while using their EURM is because of lost product keys. Removing the product key window and automatically activating the operating system for valid customers greatly reduces complexity, which in turn, increases the media's usability score, and greatly reduces service call volumes.

Costs are measured in several different ways. The first engineering consideration is the cost to build the media in the different formats currently available, which have been previously described in Table 1. The engineering goal is to always use the cheapest media format possible, which often means looking for ways to reduce the solution size to fit onto lower cost media formats. The solution size is measured to be

the entire solution package size when an International Organization for Standardization 9660 DVD image [3] (ISO image) is generated.

The second engineering consideration is to reduce the number of parts required to successfully support the platform that the EURM articles are being created for. This consideration requires that the solution be looked at from the perspective of the project on the whole, versus as an individual solution for each operating system EURM being manufactured. As an example, the platform requiring the media solution might need EURM for 4 different operating system store-keeping units (SKUs) and 2 architectures, in 30 different languages. If one article of media is created following standard Microsoft installation media format for each version of Windows the platform supports, this engineering endeavor would require the creation of 4 SKUs (Basic, Premium, Business, and Ultimate) x 2 architectures (x86 and x64) x 30 languages = 240 articles of media. This would have a devastating effect on procurement image management, forecast planning, factory space requirements, server storage space, and engineering build and test durations. As a cost example, imagine if reducing the part complexity by 1 saves the business $10,000 a year. Now, image if we reduce the number of articles of EURM required from 240 down to only 80, then we would be reducing the number of parts being managed by 160 articles; 160 x $10,000 = $1,600,000 a year in savings.

The cost savings is huge, so the engineers must look to combine as much as is logistically and legally possible. In order to facilitate this, a number of EURM parts produced matrix should be maintained and reviewed.

Operating System	# EURM	# Languages	# SKUs	# MS ISOs	Reduction
Windows 7	82	32	9	288	3.5
Windows Vista	41	25	5	125	3.05
Windows XP	50	25	2	50	1

Table 3: Media Reduction Rate

Some secondary goals to be considered when creating EURM media are build durations, test durations, server impact, and installation time.

Build durations are used for scheduling purposes and can be measured by timing the duration it takes to successfully build the media following a specific process. The somewhat natural process taken during EURM research and development is to outline the marketing expected results and then manually working with the bits to come up with a viable solution. The working process is then documented and automated. Once automated, the build durations become stable and the process can be reviewed to determine faster methods to accomplish specific tasks. In conclusion, it is an invaluable step to add time stamping routines to the developed automated build process. Illustration 4 shows sample timing data for the creation of a multi-lingual Windows 7 Service Pack 1 EURM.

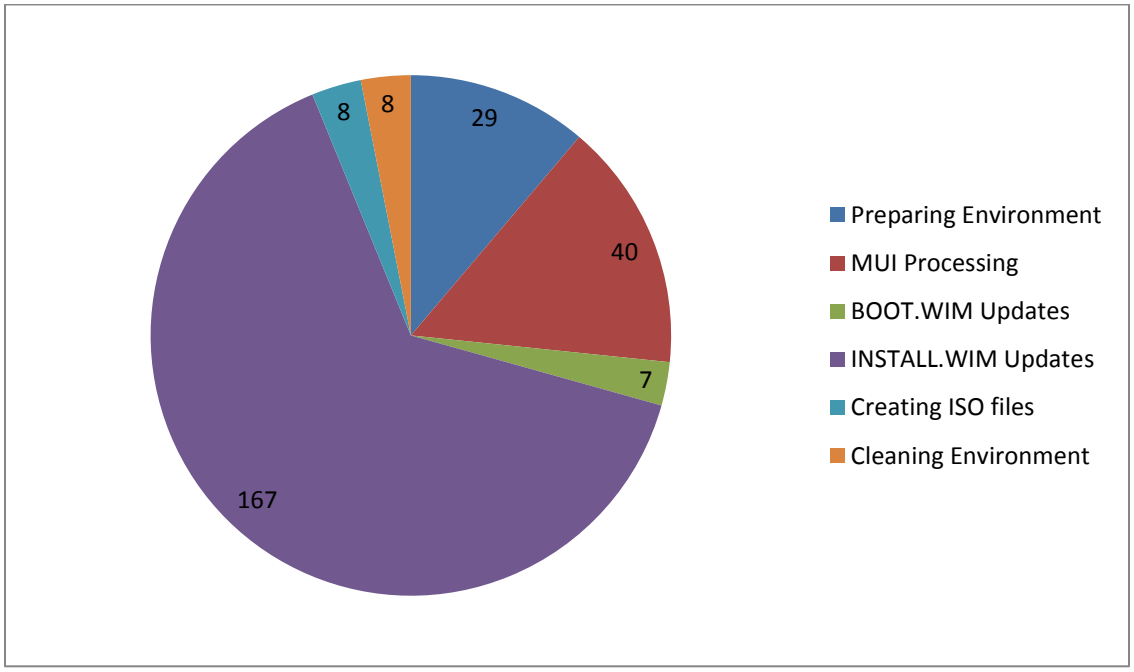

Illustration 4: Windows 7 EURM Build Duration (in minutes)

Test durations are also used for scheduling purposes and are measured by calculating the amount of time it takes to follow a documented or planned unit test. Unit test documents can also be broken down into an established set of tasks and the amount of time it takes to complete each of those task sets should be recorded and analyzed to help in process improvement discoveries. It is often possible to automate many of the test routines. Illustration 5 shows the current set of tasks that are performed during a unit test along with the amount of time (in seconds) it takes to complete each of those sets.

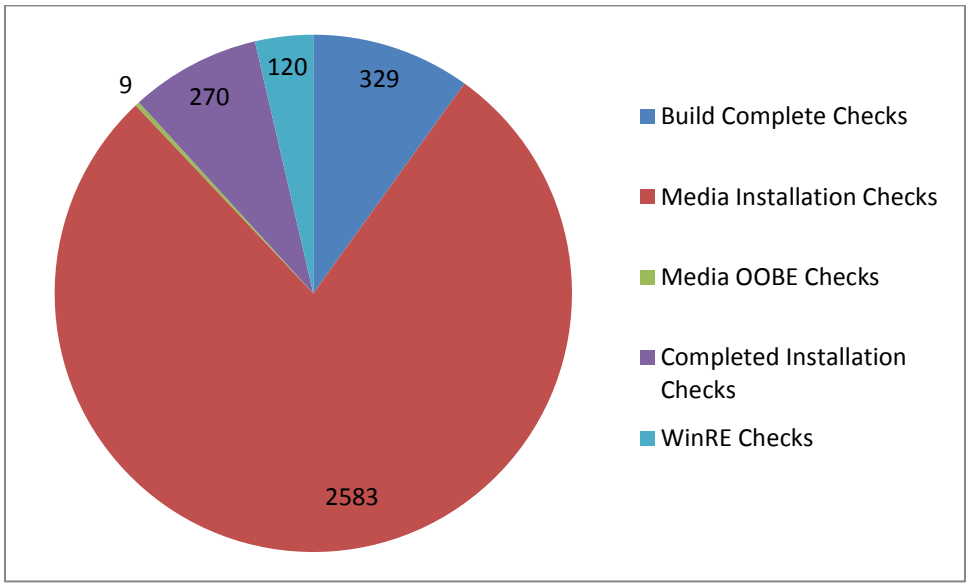

Illustration 5: EURM Test Durations (in seconds)

Server impact is the amount of server storage space the media ISO files will use when the builds are complete and the ISOs are uploaded for system integration testing and/or software procurement distribution purposes. Server impact is often studied in cases where server space is at a premium. Internal storage teams will often request server impact information for infrastructure scoping purposes. As a secondary goal, the engineering team often considers methods of reducing server impact when building and distributing their completed solutions.

Installation time is the amount of time it takes for an end user to install the operating system to a specified target system following a normal, documented procedure. This measurement can often be gathered during media unit tests. Although, installation time is not a primary goal, if the engineer can make process improvements to the installation time, then they also effect test durations and end user customer satisfaction.

Field Service Reinstallation Media (FSRM) Measurements

The primary goal for FSRM solutions is to install the selected operating system onto the customer's system using the fastest methods possible. The intent is to reduce onsite cycle time so that the field engineer can perform more customer visits in a given period of time. Some questions that need to be answered are, "*how do we measure the solution's install speed?*", and "*which media formats install which operating systems the fastest?*" [1].

Installation time is the amount of time it takes for the field engineer to install a specific operating system onto a newly installed hard disk drive. This measure can be affected by the type of media used (DVD versus USB) to deliver the operating system or the underlying technology being used. As a technology example, the installation times of applying a Microsoft image versus applying a Symantec Ghost image should be considered.

Some secondary goals for the FSRM solution are to design the product to be sustainable, easy to use, and applicable to the systems being serviced.

Sustainability is the measure of how easily this solution can be updated when issues are found with the product or product improvements are discovered, engineered, and ready to roll-out. This is a big deal for the FSRM solutions because the organization will often have field engineers stationed all over the world. Making product updates in

this type of scenario is often hard to facilitate without a good mechanism being available along with a solid solution that can be quickly updated or modified on-the-fly in the field.

Ease of Use is the measure of how long it takes for the field engineer to learn to use the FSRM solution. Overly complicated solutions can lead the engineer to make mistakes during their onsite customer visits, which can affect both the installation time and the customer satisfaction ratings.

Applicability is the measure of how well the solution can cover the scope of products the field engineer is chartered to support. If the solution will not work on specific systems, then the solution is then limited in its scope of usefulness. Illustration 6 is a sample chart that might be developed to show that using the USB format gives a greater range of applicability to the field engineer since it supports a wider range of platforms the engineer might be required to support.

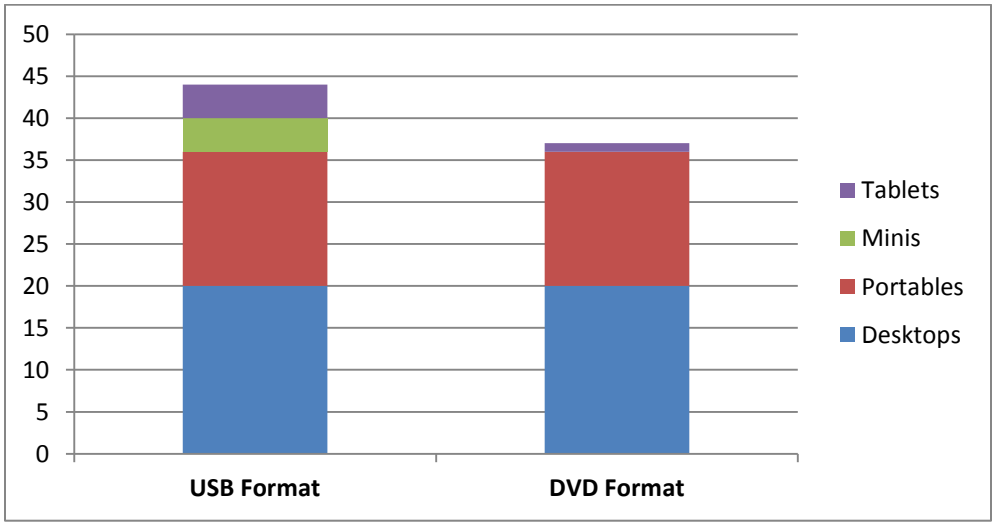

Illustration 6: Applicability Chart (example)

Chapter 4: Removable Media Improvements

Focusing on the primary and secondary goals of removable media solutions requires the engineering team to first determine how and what to measure, as shown in Chapter 3: Removable Media Measurements. Once the engineering team gathers appropriate measurements they can then analyze them for possible process improvements.

End User Reinstallation Media (EURM) - Unit Test Improvement

The EURM unit test (Illustration 5 from Chapter 3), had several task sets that can be analyzed for possible process improvements. The most obvious improvement that can be made is to automate some of the tasks within each task set in an attempt to decrease the time it takes to complete a unit test. With EURM media, this isn't necessarily the easiest thing to do, especially since the solution is often designed to run on specific systems with specific BIOS configurations, and a direct connection to these systems is often required in order to successfully complete a unit test.

Our first step when reviewing the unit test results is to brainstorm ideas on how we might make each of the tasks faster. After we have completed our analysis, we can then decide which ideas we believe are the easiest to complete with the biggest impact towards our goal of reducing the test duration time requirements.

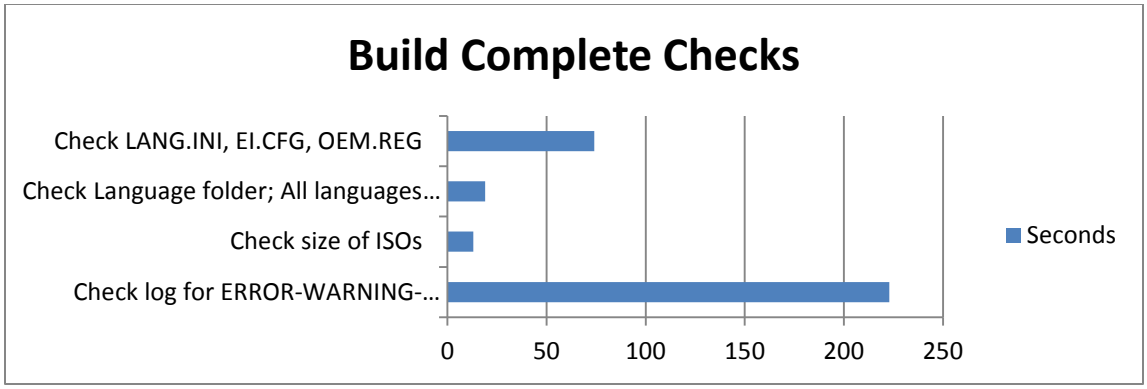

Illustration 7: Build Complete Checks

Illustration 7 is a break-down of the individual tasks from the ***Build Complete Checks*** task set, previously shown in Illustration 5 of Chapter 3. One of the longest steps recorded was the task to *check log for error-warnings-failures*. This task seems easy to automate and would have a sizable positive impact towards reducing the unit test duration. Since most projects have set schedules that they need to be completed by, we often focus on the items that will have the biggest impact first, leaving the lowest impact items to be done last, as time permits.

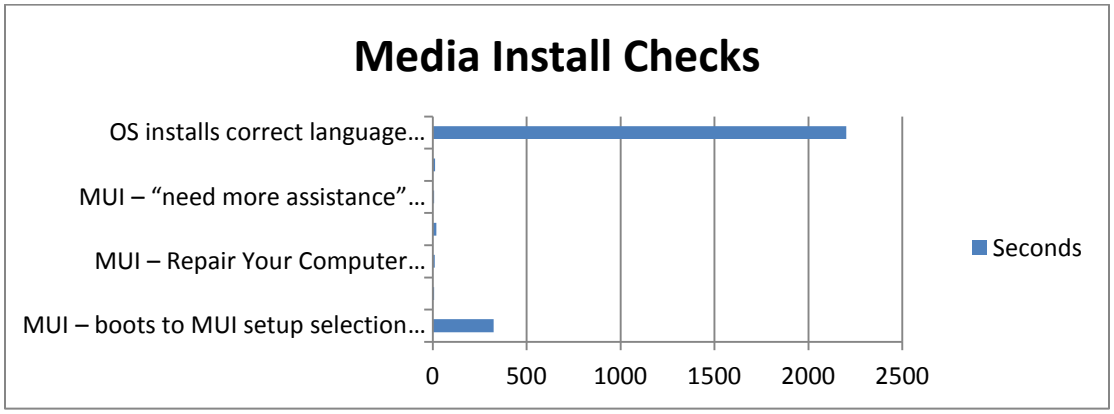

Illustration 8: Media Install Checks

Illustration 8 shows the break-down of the longest task set, the ***Media Install Checks***, previously described in Illustration 5 of Chapter 3. This task set would appear to be the most obvious task to review with the most scrutiny, since it adds the most time

to the entire process. However, when the individual tasks are reviewed, it becomes clear that the longest task, "*OS installs correct language…"* might not gain much improvement with regards to test durations, even if the process is automated. This is the step where the image is extracted from the media solution onto the target hard disk drive. So, even if the test task were to be automated, it might still take the same amount of time to complete. This task is thus considered to be a low impact item to automate for unit test. That said, one of the lower priority secondary goals is to increase installation velocity and there is always the benefit of being able to launch a test to run overnight if the task were to be automated, therefore, this step might be further scrutinized to determine if there are faster ways to apply the image. To be clear, automating the test task might not give much impact towards decreasing the test duration, but re-engineering the process used to apply the image might help in both decreasing install time and reducing test durations. However, the new solution might be very complicated and would greatly increase risk to the project if new, unproven, methods are utilized.

For this project, one of our higher priorities is to reduce build and test durations, so we first decide to automate the *check log* task in the ***build complete checks*** test set, by creating a parsing program that will read each line of the output logs to determine if any errors, warnings, or failures are present. If any suspect messages are found, they can then be presented to the engineer in a format that helps the engineer quickly find the message in the log to help with trouble shooting routines.

To determine appropriate strings to parse for, we first force failures for the following cases:

- **DISM Failure** – Microsoft created a tool called *Deployment Image Servicing and Management* (DISM) [2] that can be used to update an operating system image while it is offline. During the media build process we use DISM to inject language

packs, local packs, and Microsoft Windows updates. We need to know if any of these calls failed during the build process. A forced applicability error generated a line as shown below:

Error: 0x800f081e

So, we decided to search for any instances of "**Error:**" in our error parsing program.

- **File copy errors** – for explicit file copy commands, a response will be generated if the file cannot be found during the build process. Our forced error appears as:

  File not found - ei.cfg

  So, we will search for the string, "**File not found**" in our error parsing program.

- **File delete errors** – for file delete/move errors, we have two possible bad outcomes, one is if the file does not exist before we try to delete/move it and the other is if the file cannot be deleted because it is in use or we do not have the correct permissions. We generated the following errors as references for our error parsing program:

  The process cannot access the file because it is being used by another process.

  Could Not Find C:\Win7sp1Media\vssver.scc

  Based on these possible issue results, we will search for the strings, "**Could Not Find**", and "**cannot access**".

- **ImageX failure –** ImageX [2] is a Microsoft tool that allows OEMs to capture and modify images. For our scenario, this tool can fail if the image file does not exist or if the target directory is in use or if the target directory does not exist or if the image is already mounted on the same volume. We forced those errors to produce the following errors listed below:

Error opening file [c:\w7sp1\inst.wim].

Error mounting image.

Error setting temp path: [c:\mount].

[WARN] An objectID is in use on this volume

Based on these results, we modify the previous search entry of "**Error:**" to be "**Error**". To cover the case of an image already being mounted, we search for the "**WARN**" message.

- **Intlcfg Failure –** Intlcfg [2] is a Microsoft tool that we use to automate the creation of a lang.ini file and preset the default localization settings to an image offline. Some suspected failure samples are generated for when a mounted image does not exist, is not writable, or incorrect localization parameter is used. We generated these failures to script against:

  ERROR: An invalid path specified.

  ERROR: An invalid path specified.

  ERROR: API call failed with following error(s):

  Based on these results, we can see that we are already covered by our earlier defined "**Error**" search parameter; however, we do need to ensure that our program is not case sensitive.

- **Oscdimg failure –** Oscdimg [2] is a Microsoft tool used to generate an ISO image of a specified directory. We use this tool to generate the DVD and USB media ISOs that are then distributed to test teams and procurement. Some sample problems that might arise when using this tool during our process are missing boot image, using an invalid source path, or using an invalid output path. Therefore we generated errors for each of the instances listed below:

ERROR: Could not open boot sector file

ERROR: Failure enumerating files in directory

Error 3: The system cannot find the path specified.

Based on these results, we can see that we are already covered by our earlier defined "**Error**" search parameter.

Now that we have a list of failure strings to parse for (**Error**, **WARN**, **Could Not Find**, **cannot access**, **File not found**), we design our testing solution to take text based input, parse the input for failure strings, and if found, display the line number and the failure message. Our tool should continue processing the text until all lines have been read and all errors have been reported. To verify the functionality of our tool, we should, at a minimum, verify it correctly identifies all of the failure messages listed in the previous paragraphs. We should also document any false positives to help determine if we need to update our build process, update the tool with some branch filters, or just leave them alone as noted false positives.

Illustration 9 shows the output of our newly created log checker after it has been run against a sample media build log of Windows 7 SP1 Beta. With line numbers as reference points, we can open the build log in Notepad ++ (from http://notepad-plus-plus.org/) and quickly find the error message and its surrounding entries.

The first entry in our error log shows:

LINE 172: Could Not Find C:\Win7sp1Media\vssver.scc

After reviewing the build log, we determined that this was a step in the build process that cleans Visual Source Safe remnants from the resultant customer media.

```
LINE 172: Could Not Find C:\Win7sp1Media\vssver.scc
LINE 1286: Error: 0x800f081e
LINE 1343: Error: 0x800f081e
LINE 1404: Error: 0x800f081e
LINE 1441: Error: 0x800f081e
LINE 1518: Error: 0x800f081e
LINE 1575: Error: 0x800f081e
LINE 1636: Error: 0x800f081e
LINE 1673: Error: 0x800f081e
LINE 1750: Error: 0x800f081e
LINE 1807: Error: 0x800f081e
LINE 1868: Error: 0x800f081e
LINE 1905: Error: 0x800f081e
LINE 1986: Error: 0x800f081e
LINE 2043: Error: 0x800f081e
LINE 2104: Error: 0x800f081e
LINE 2141: Error: 0x800f081e
LINE 2220: Error: 0x800f081e
LINE 2277: Error: 0x800f081e
LINE 2338: Error: 0x800f081e
LINE 2375: Error: 0x800f081e
LINE 4362: Error: 0x800f081e
LINE 4802: Error: 0x800f081e
LINE 4819: Error: 0x800f081e
LINE 5277: Error: 0x800f081e
LINE 5715: Error: 0x800f081e
LINE 5732: Error: 0x800f081e
LINE 6189: Error: 0x800f081e
LINE 6628: Error: 0x800f081e
LINE 6645: Error: 0x800f081e
LINE 7101: Error: 0x800f081e
LINE 7540: Error: 0x800f081e
LINE 7557: Error: 0x800f081e
LINE 8014: Error: 0x800f081e
LINE 8426: File not found - ei.cfg
LINE 8510: File not found - ei.cfg
```

Illustration 9: Automating Log Review

We considered this to be a false-positive result, even though the message “Could Not Find” was discovered; the message is not considered to be a real issue since the file does not exist prior to the media build call to delete it – so in reality, it is good that the file does not exist, which is what we want. We can approach this scenario in several ways, correct the build process to check for the existence of the file before deleting it, note it as a false-positive in our release notes, or change the error checking routine to ignore “Could Not Find” messages that reference vssver.scc files. In this case, since we are still in our media build process development phase, we opted to update the build process to check for the existence of the file before attempting to delete it.

Our second issue of note was:

LINE 1286: Error: 0x800f081e

When looking up line 1286 in our build log and reviewing the error in context with its preceding and following lines, we discover this issue is due to a DISM call that is attempting to install a language pack, see Illustration 10.

```
DISM /image=c:\boot_mount_x64 /Add-Package /PackagePath:c:\win7media\WPE\amd64\de-de\lp.cab

Deployment Image Servicing and Management tool

Version: 6.1.7600.16385

Image Version: 6.1.7601.17077

Processing 1 of 1 - Adding package Microsoft-Windows-WinPE-LanguagePack-
Package~31bf3856ad364e35~amd64~de-DE~6.1.7600.16385

Error: 0x800f081e

The specified package is not applicable to this image.
```

Illustration 10: DISM Failure

We discover the DISM call fails its applicability rules because the language pack is designed for Windows 7 SP0, and our process is attempting to install it onto a Windows 7 SP1 image. This is a true issue and is something we want our error checking routine to look for. In this scenario, we were attempting to integrate German, Japanese, French, and Spanish into our English media, and discovered that all of our language pack integration attempts were failing. Line 1343 through Line 2375 were all related to failed language pack integration attempts.

From Illustration 9, the errors reported from Lines 4362 through Line 8014 were also DISM applicability errors. Though similar to the language pack install issues, these errors were actually caused when attempting to integrate Windows 7 SP0 Microsoft Windows updates into the Windows 7 SP1 image. This is another true issue, which we would want to know about. After replacing the language packs and Microsoft updates with appropriate versions, these issues should go away.

Line 8426 and Line 8510 were also real issues because we were attempting to copy files that do not exist. After reviewing our process, we discover that our "test" media is constructed differently than the media we plan to ship to our end users. Our test media includes all SKUs, so the tester can select which operating system flavor they want to install and test while our true customer media will only contain one SKU. For this issue, we decide to add a check to the build process to only copy the ei.cfg file if the media build is determined to be non-test media.

The last step we take is to integrate the log check program into our build process to help reduce our unit test durations. We will then rerun our media build routine, collecting timing data to help us determine how successful we were in reducing our unit test durations.

Field Service Reinstallation Media (FSRM) – Install Speed Improvement

The most important aspect of our FSRM solution is to install an operating system in the fastest manner possible. [1] This requirement greatly outweighs all other concerns. While reviewing our current FSRM USB solution, we suspected it was taking longer to install Windows 7, than if we were to install the same operating system using our end user reinstallation media (EURM) DVD. In order to validate this suspicion, we first gathered install times of both the FSRM USB media and the EURM DVD media and compared them.

We decided to use a virtual machine to compare the two media install concepts by attaching their images as a DVD drive. In theory, this would allow us to remove USB and DVD transfer speeds from the equation so that we could focus our attention on the underlying software architecture. We approached this study by first timing the EURM DVD ISO install of English Windows 7 using a virtual machine and recording the times for the "boot startup" phase, the "menu selections" phase and then the "image extraction" phase. We then attempted to time the FSRM USB ISO on the same virtual machine.

When running the FSRM USB ISO we ran into an unexpected issue. Our virtual machine would "lock up" with a blinking cursor in the top left corner of the screen during startup whenever the FSRM USB ISO was attached to the system (as shown in Illustration 11).

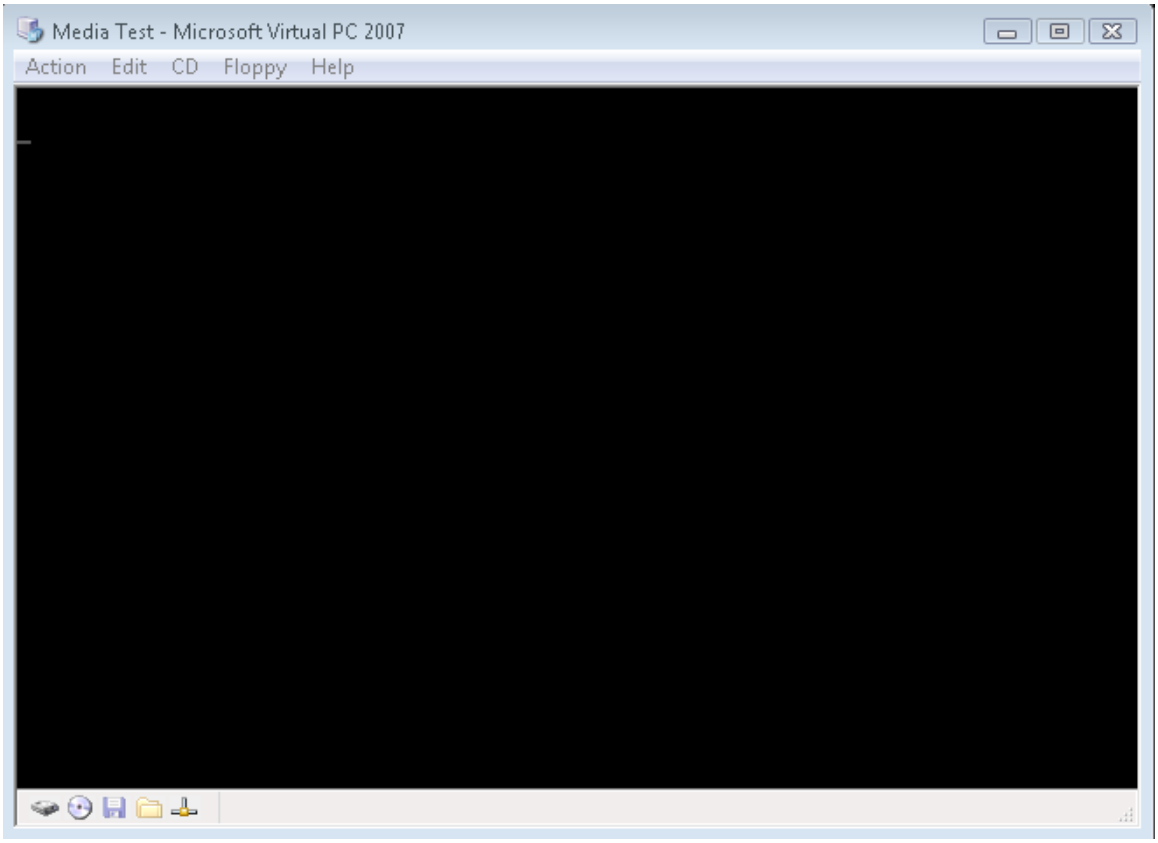

Illustration 11: Virtual PC - blinking cursor issue

Our first thought was that the ISO must be too big for the virtual machine to handle. So, we retrieved a dual-layered multi-lingual DVD ISO and attempted to launch the virtual machine with the large ISO attached. It ran successfully, with no issues, indicating to us that the virtual machine drivers supported large DVD formats. We then reviewed the ISO-9660 standards and determined that file names of more than 30 characters are not supported. When reviewing our field drive, we discovered that it contained an image with a name that was 32 characters long, so, for the sake of our

timing study, we renamed the image file to be “install.wim”, which is only 11 characters long. After regenerating our FSRM ISO and attempting to boot into the virtual machine we discovered that this did not fix the issue. We also double-confirmed that long file names should not be an issue by reviewing the help information provided with the ISO creation tool we were using (OSCDIMG.exe) [2]. Illustration 12 shows the OSCDIMG command we were using to generate our ISOs:

```
OSCDIMG –e –h –m –u2 –lFieldDrive –bc:\field\boot\etfsboot.com c:\field c:\iso\fieldTest2.iso
```

Illustration 12: OSCDIMG command line

According to information found in the OSCDIMG “-help UDF” command, the “u2” parameter forces the ISO we are creating to use the UDF file system instead of the ISO9660 file system. The UDF file system allows for long filenames, thus ruling out our long file name theory as a possible cause of our issue.

We now decided to take our working Windows 7 DVD ISO and extract its contents into a work folder and then recapture those contents using our OSCDIMG command. Our new train of thought was that maybe our tool or tool parameters were causing the ISO to become unreadable to the virtual machine. After the new ISO was created, we again attempted to boot into the virtual machine with our attached ISO, however, this time everything ran successfully, with no issues. This now increased our confidence in our ISO creation tool and the parameters we were using.

Now, since we knew the work folder contents were working, we removed the EURM image “sources\install.wim” and added our FSRM image as images\install.wim. After rebuilding and attempting to run our new ISO, the issue once again presented itself. This gave us a strong indication that there might be something fundamentally wrong with the image we were attempting to embed into the ISO, at least as far as the virtual machine was concerned. As a final test, we decided to move our

images\install.wim file to sources\install.wim (this is our field drive install.wim file not the EURM version) and then we recaptured and tested the ISO. Strangely, our virtual machine now has no issues with the image. So, we now had a possible workaround that would enable our field drive ISO to attach and run in the same virtual machine that we performed our EURM time study with.

Our next step was to update our image extraction logic to recognize the new path and file name and we then generated a new FSRM USB ISO. Since this ISO now runs in our virtual machine, we were finally able to gather appropriate measurements that would allow us to compare the EURM DVD ISO install times to the FSRM USB ISO install times.

	EURM DVD ISO	FSRM USB ISO
Boot Startup	49 sec	55 sec
Menu Selections	49 sec	18 sec
Image Extraction	9 min 01 sec	33 min 16 sec

Table 4: Installation Timing

Our timing data confirmed our suspicion that the FSRM USB ISO media was taking longer to install than the EURM DVD ISO, and since we used a virtual machine to perform the test, we know the issue is related to our solution architecture rather than DVD versus USB bit transfer speeds.

Since the biggest hit against the installation duration was with the image extraction phase, we decided to focus our attention on that routine to determine where we were going wrong. We found that the current FSRM solution installed a MUI operating system (containing English, German, Italian, and French) that allows the customer to choose their language at first boot rather than a single language operating system. The original architecture was attempting to reduce menu selection complexity by eliminating the need for the field service engineer to select an operating system

language. By always installing a MUI OS, the field service engineer only needed to select the operating system SKU and the process would run. This however installed a much larger operating system image, which took more than 20 minutes longer to install than if the engineer were forced to select a language and install a smaller, single language operating system.

Therefore, we proposed to improve the process by adding an additional step to the menu selection process and then install only single language operating system instances. After making the changes, we can then gather new installation times to determine if our new architecture would have any effect on our FSRM media installation times.

Chapter 5: Removable Media Analysis

After process improvements are implemented, we then collect measurements on those improvements to determine if they are having a positive or negative effect on the process as a whole.

End User Reinstallation Media (EURM) - Unit Test Analysis

After automating the task of *reviewing build logs for error messages* we now gather new metrics and compare the results to the previous durations (Illustration 13).

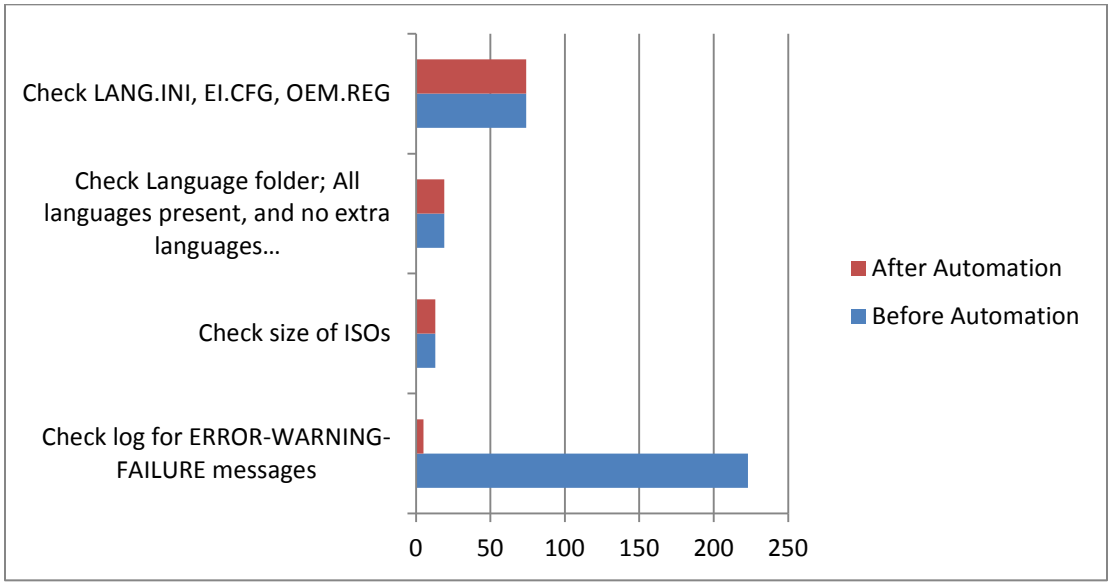

Illustration 13: Automation Analysis

The only task that we automated was the parsing of the log for error messages, and there was a substantial improvement. When physically reviewing the build logs for error messages, it took the engineer 223 seconds to complete the task for one article of media. When the step was automated, it took the computer 2 seconds to complete the task and it took the engineer 3 seconds to locate the resultant error check log and open it to determine if any issues were found. This has reduced the unit test by 218 seconds (223 – 5) per article of media being produced. A typical operating system launch effort

of 82 articles of media (see Table 3) means we are already reducing media build durations by 17,876 seconds (82 articles of media x 218 seconds each), or approximately 5 hours.

Another potential benefit of automating the task, which is not captured in the reduced build time metric, is that we might also have less misses by exhausted engineers. We believe the computer will catch more issues than the engineers that are required to manually scan the build logs for issues. In order to prove this, we would have to count the number of misses the engineers have had in the past and compare it to the number of misses we are having after this step was automated. We currently don't track engineering misses that could have been caught if the engineer had reviewed the build log more accurately, so, we can only speculate at this time.

Field Service Reinstallation Media (FSRM) – Install Speed Analysis

	EURM DVD ISO	FSRM USB ISO	Updated FSRM
Boot Startup	49 sec	55 sec	54 sec
Menu Selections	49 sec	18 sec	19 sec
Image Extraction	9 min 01 sec	33 min 16 sec	7 min 54 sec

Table 5: Installation Timing Improvement

Our updated FSRM process proved to have much faster installation results when compared to the previous FSRM process. Although the menu selection process had an additional bubble selection menu the field engineer had to select a language from, it only added one second for the engineer to process. The image extraction phase had a substantial improvement and overall, the FSRM USB solution now installs images slightly faster than the EURM DVD solution.

After gathering measurements on the durations to deploy images in the field and then weighing the goals of the product, we decided to for-go the one second faster, less complicated, menu selection route in order to reduce overall onsite visit cycle times.

The new menu selection routine is no more complicated than what is used with the Windows XP operating systems, so we predict that there will be no dramatic effect on the ease-of-use factor for the field service engineers; thus requiring no additional training.

Chapter 6: Conclusion

Overall, the key to software engineering removable media is to first define the goals of the project and then develop proper measurement techniques to help ensure we are doing everything we can to accomplish our goals.

With the end user reinstallation media, we reviewed our goals to determine what sort of process improvements we believed we could benefit from most before taking action. We paid attention to our primary goal while attempting to improve one of our secondary goals, which ensured the end result was of actual benefit to the project on the whole.

With the field service reinstallation media, we also found a way to improve our goals through careful measurement analysis of the software architecture. Although in the field, it might sometimes be necessary to compare installation times with a stop watch (comparing DVD to USB) installations, it is hard to determine if there can be improvements made to the underlying software architecture if there are large differences in hardware to hardware transfer rates. Using the virtual machine as our controlled environment, we were able to determine that USB drives were deploying Windows images much slower than DVD media, and root caused it to a previous process improvement of making the menu selection process of OS architecture and language more simple. Staying focused on our primary objectives, we decided to increase complexity to save a large amount of time in install velocity.

While working on both removable media projects we realized the importance of measuring results and staying focused on each of our project's primary goals. This will allow us to make ongoing improvements without jeopardizing our main objectives.

Bibliography

[1] Zabava, B. (2010). *Measuring Removable Media Solutions* (Unpublished EE N382V report). University of Texas at Austin, Austin, TX.

[2] Windows OEM Preinstallation Kit (Windows OPK) User's Guide for Windows 7. (2009). Microsoft. Included with Windows 7 OPK software, build date: 2009-06-04. http://www.microsoft.com/oem/en/downloads/Pages/windows_7_opk.aspx

[3] How to burn an ISO image. (2007). CentOS. Johnny Hughes. Retrieved September, 2010, from http://www.centos.org/docs/5/html/CD_burning_howto.html